

# A Linear-Time Algorithm for Trust Region Problems

Elad Hazan

Technion

ehazan@ie.technion.ac.il

Tomer Koren

Technion

tomerk@technion.ac.il

January 28, 2014

## Abstract

We consider the fundamental problem of maximizing a general quadratic function over an ellipsoidal domain, also known as the trust region problem. We give the first provable linear-time (in the number of non-zero entries of the input) algorithm for approximately solving this problem. Specifically, our algorithm returns an  $\epsilon$ -approximate solution in time  $\tilde{O}(N/\sqrt{\epsilon})$ , where  $N$  is the number of non-zero entries in the input. This matches the runtime of Nesterov's accelerated gradient descent, suitable for the special case in which the quadratic function is concave, and the runtime of the Lanczos method which is applicable when the problem is purely quadratic.

## 1 Introduction

Perhaps the most elementary quadratic optimization problem is that of maximizing a quadratic function over the unit ball. This is precisely the *trust region* problem, or formally:

$$\begin{aligned} & \text{maximize} && x^\top Ax + 2b^\top x \\ & \text{subject to} && \|x\|^2 \leq 1, \end{aligned} \tag{1}$$

where  $A \in \mathbb{R}^{n \times n}$  is an arbitrary  $n \times n$  symmetric (possibly indefinite) matrix and  $b \in \mathbb{R}^n$ .

The trust region problem has numerous applications in optimization, where trust region methods (Conn et al., 2000) are among the most empirically successful techniques for solving nonlinear optimization problems. In these methods, that enjoy strong convergence properties, at each iteration of the algorithm a quadratic approximation of the objective function is minimized over a ball, called the trust region. Trust region problems are also useful in combinatorial optimization (Busygin, 2006), least-squares problems (Zhang et al., 2010), constrained eigenvalue problems (Gander et al., 1989), and more.

Despite being non-convex, the trust region problem has been shown to exhibit strong duality properties and is known to be solvable in polynomial time (e.g., Ben-Tal and Teboulle 1996; Ye and Zhang 2003). More specifically, it can be shown to be equivalent to more complex, albeit convex, semidefinite programming (SDP) optimization problems that can

be solved using interior-point methods in polynomial time (Nesterov et al., 1994; Alizadeh, 1995). However, the worst-case complexity of current interior-point based solvers for SDP problems is a large polynomial, and they are inefficient in exploiting sparsity of the data. As a consequence, this approach is not practical for large-scale problems.

On the other hand, the close connections between the trust region problem and eigenvalue problems suggest that more efficient trust region algorithms should exist. Indeed, if the problem is purely quadratic, i.e., when  $b = 0$ , then the trust region problem reduces to the fundamental maximal eigenvector problem, that can be approximated in linear time via the well-known Power and Lanczos methods. This observation has led authors in the optimization and the numerical analysis communities to develop efficient, matrix-free algorithms that are based solely on matrix-vector products. Notable examples are the dual-based algorithms of Moré and Sorensen (1983), Rendl and Wolkowicz (1997) and Sorensen (1997), the generalized Lanczos trust-region method of Gould et al. (1999), and the more modern advancements of Rojas et al. (2001); Erway and Gill (2009); Erway et al. (2009); Gould et al. (2010). However, while being provably convergent in most cases, the runtime evaluation of these algorithms is essentially empirical and lacks formal guarantees. To the best of our knowledge, to date there is no formal evidence that the trust region problem can be solved in linear time (in the worst case), as the closely-related maximal eigenvalue problem.

The main hurdle faced by most previous methods is a certain case in which numerical difficulties arise, so-called the “hard case” (Moré and Sorensen, 1983), and most research in the last two decades on the trust region problem focuses on addressing this issue. This phenomenon occurs when the linear component vector  $b$  is nearly orthogonal to the eigenspace of the smallest eigenvalue of  $A$ , and seems to be the reason for the lack of provable worst-case convergence bounds for the trust region problem.

**Our contribution.** In this paper we show that the additional linear term and the non-convexity of the problem do not add to its complexity: we devise a novel linear-time algorithm for approximating the trust region problem that has, up to logarithmic terms, the same worst-case time complexity of *a single maximal eigenvalue computation*. Our approach reduces the trust region problem into a series of eigenvalue computations, thus it is able to exploit data sparsity and runs in time linear in the number of non-zero entries of the input. In the specific case where the problem is convex (i.e., when  $A$  is negative semidefinite), the same complexity guarantees can be obtained by applying Nesterov’s accelerated gradient descent (Nesterov, 1983) to the problem; thus, our approach can be seen as an analog of the latter algorithm to the general non-convex case.

Our approach is based on an SDP relaxation of a feasibility version of the trust region problem. While SDP relaxations are already standard for this problem (e.g., Rendl and Wolkowicz 1997; Ye and Zhang 2003), our approach uses a specific form of SDP that can be approximated quickly via eigenvalue computations and does not require us to use interior-point solvers. Another important feature of our relaxation is that it admits an efficient and accurate rounding procedure; that is, given a matrix solution to the SDP we are able to extract an approximate vector solution to the original trust region problem of almost the same qual-

ity. This rounding procedure allows us to avoid numerical problems and complications that exist in previous approaches, which occur for certain configurations of the eigenvalues of  $A$ , namely the “hard case” mentioned above.

At the heart of our approach is an efficient, linear-time solver for the SDP relaxation. This solver exploits the special structure of the dual problem, which is essentially a one-dimensional problem for which bisection techniques can be applied. Each dual step of this algorithm, that converges in a logarithmic number of iterations, amounts to a single approximate eigenvalue computation. The major technical difficulty that results from working in the dual, is in obtaining a primal solution from the dual iterations. To this end, we employ a technique reminiscent of the “ellipsoid against hope” algorithm (Papadimitriou and Roughgarden, 2008), used by Arora et al. (2005) in the context of approximate semi-definite programming, for recovering a primal solution by solving a small linear program formed by the dual iterates.

## 2 Setup and Statement of Results

In this section we formalize the setting and state our main results. We shall describe our algorithm and results in a slightly more general setting, that include optimization problems of the form:

$$\begin{aligned} & \text{maximize} && x^\top Ax + 2b^\top x \\ & \text{subject to} && \|x\|_M^2 \leq 1, \end{aligned} \tag{2}$$

in which the optimization domain is an ellipsoidal set, described by a general norm constraint  $\|x\|_M \leq 1$ . Here,  $\|x\|_M = \sqrt{x^\top Mx}$  is a norm induced by a positive definite matrix  $M \in \mathbb{R}^{n \times n}$ . We note that the optimal solution  $v^*$  to this problem is necessarily non-negative, as the objective function equals zero for  $x = 0$  and the value at the optimum can only be larger.

For our bounds, we use the following notation. We let

$$\begin{aligned} \lambda &= \max \{2(\|A\|_2 + \|b\|), \|M\|_2, 1\} , \\ \mu &= \min \{\lambda_{\min}(M), 1\} , \end{aligned}$$

where  $\|\cdot\|$  is the Euclidean vector norm, the matrix norm  $\|\cdot\|_2$  is the spectral norm, and  $\lambda_{\min}(\cdot)$  and  $\lambda_{\max}(\cdot)$  refer to the minimal and maximal eigenvalues of a matrix. We refer to the ratio  $\kappa = \lambda/\mu$  as the “condition number” of the problem<sup>1</sup>. The objective is  $\lambda$ -Lipschitz, so the optimal value  $v^*$  lies in the interval  $[0, \lambda]$ . For our runtime results, we also let  $N$  be an upper bound over the number of non-zero entries in the matrices  $A$  and  $M$ , and assume without loss of generality that  $N \geq n$ .

As mentioned earlier, our goal is to reduce the approximation of problem (2) in the general case to a series of approximate eigenvalue computations, and thereby obtain an algorithm that runs in time linear in the number of non-zero entries  $N$  in the matrices  $A$  and  $M$ . To this end, we formally define the notion of an approximate eigenvalue oracle.

---

<sup>1</sup>Notice that this condition number can be approximated in linear time (see Section 4.1).

**Definition 1.** *An approximate eigenvalue oracle is a randomized procedure that given a matrix  $A \in \mathbb{R}^{n \times n}$  and parameters  $\epsilon, \delta > 0$ , with probability at least  $1 - \delta$  returns a vector  $x \in \mathbb{R}^n$  such that  $x^\top Ax \geq \lambda_{\max}(A) - \epsilon$ .*

An approximate eigenvalue oracle can be implemented to run in linear time via the Lanczos method. For completeness, this is formally shown in Section 4.1.

Our algorithm solves the equivalent feasibility problem

$$\begin{aligned} x^\top Ax + 2b^\top x &\geq c + \epsilon/\kappa \\ \|x\|_M^2 &\leq 1 - \epsilon/\kappa, \end{aligned} \tag{3}$$

with  $c \in [0, \lambda]$ . Indeed, we can reduce the task of  $\epsilon$ -approximating the trust region problem to (3) via a binary search over the optimum  $v^*$  (incurring an additional log-factor to the running time), with  $c$  being our current guess of  $v^*$ . Since  $\epsilon/\kappa \leq \epsilon$ , tightening the first constraint by  $\epsilon/\kappa$  only harms the approximation by a constant factor. Note that we have also relaxed the second constraint by  $\epsilon/\kappa$ , but this only shifts the optimal solution by an Euclidean distance of at most  $(1/\mu)\epsilon/\kappa$ , which in turn translates to an additional  $(\lambda/\mu)\epsilon/\kappa = \epsilon$  bias in the objective value (since the objective function is  $\lambda$ -Lipschitz).

More specifically, the algorithm we describe approximately solves (3), in the sense that it either correctly declares that the problem is infeasible or finds a vector  $x \in \mathbb{R}^n$  such that

$$\begin{aligned} x^\top Ax + 2b^\top x &\geq c \\ \|x\|_M^2 &\leq 1. \end{aligned} \tag{4}$$

The main result of this manuscript is the following.

**Theorem 2.** *Given an approximate eigenvalue oracle and parameters  $\epsilon, \delta > 0$ , with probability at least  $1 - \delta$  (over the randomization of the oracle) Algorithm 1 below returns a vector  $x \in \mathbb{R}^n$  for which (4) holds or correctly declares that (3) is infeasible. The algorithm invokes the oracle  $O(\log(\kappa/\epsilon))$  times and can be implemented to run in total  $\tilde{O}(N\sqrt{\kappa/\epsilon})$  time<sup>2</sup>.*

A proof of the theorem is provided in Section 3.1.

### 3 The Algorithm

In this section we describe our linear-time algorithm for the feasibility problem (1), which is summarized in Algorithm 1.

**SDP relaxation.** The first step in our approach is to relax the feasibility problem (3) to the following SDP program:

$$\begin{aligned} A_i \bullet X &\geq \epsilon'/2 \quad (i = 1, 2), \\ X &\in \mathcal{K}_{n+1}, \end{aligned} \tag{6}$$

---

<sup>2</sup>Throughout, we use the  $\tilde{O}$  notation to hide constant and poly-logarithmic factors.

---

**Algorithm 1** TrustRegion( $A, b, M, \epsilon$ )

---

**input**  $A, M \in \mathbb{R}^{n \times n}$ ,  $b \in \mathbb{R}^n$  and  $\epsilon > 0$ **output** a vector  $x \in \mathbb{R}^n$  satisfying (4), or infeasibility of (3)1: let  $\epsilon' = \epsilon/2\kappa$ 2: define the  $(n+1) \times (n+1)$  matrices

$$A_1 = \frac{1}{2\kappa} \cdot \begin{pmatrix} -c & b^\top \\ b & A \end{pmatrix}, \quad A_2 = \frac{1}{2\kappa} \cdot \begin{pmatrix} 1 & 0 \\ 0 & -M \end{pmatrix} \quad (5)$$

3: invoke **SolveSDP**( $A_1, A_2, \epsilon'/2$ ) that returns  $X = \sum_{i=1}^r x_i x_i^\top$  as output4: **if** **SolveSDP** returned “infeasible” **then**

5:   conclude that (3) is infeasible

6: **else**7:   invoke **SZRotation**( $A_1, X$ ) that returns  $X = \sum_{i=1}^r y_i y_i^\top$  as output8:   find a vector  $y \in \{y_1, y_2, \dots, y_r\}$  for which  $y^\top A_2 y \geq \epsilon'/4r$ 9:   let  $\alpha = y(1)$  be the first entry of  $y$  and let  $\tilde{y} = (y(2), y(3), \dots, y(n+1))$ 10:   **return**  $x = \tilde{y}/\alpha$ 11: **end if**

---

with  $A_1, A_2$  given in Eq. (5),  $\epsilon' = \epsilon/\kappa$  and

$$\mathcal{K}_n = \{X \in \mathbb{R}^{n \times n} : X \succeq 0, \text{tr}(X) \leq 1\}$$

being the set of all  $n \times n$  positive semidefinite matrices with trace at most one. Here and henceforth, we use the notation  $A \bullet X = \text{trace}(A^\top X)$  to denote dot-products of matrices. To see that this is indeed a relaxation, note that given any feasible solution  $x$  to (3), the rank-one matrix  $X = \frac{1}{2}(1, x) \cdot (1, x)^\top$ , with trace  $\text{tr}(X) = \frac{1}{2}(1 + \|x\|_2^2) \leq 1$ , is feasible for the SDP.

This specific form of SDP has two important advantages. First, the structure of  $\mathcal{K}_{n+1}$  allows us to use the eigenvalue oracle for solving the SDP, as a linear optimization over this set is equivalent to a maximal eigenvalue computation. Second, as we show in our analysis, the artificial extra slack of  $\epsilon'/2$  we imposed in the constraints ensures that a solution to this relaxation has sufficient mass on its first entry, which makes it possible to avoid the “hard case” entirely.

**Linear-time SDP solver.** The main ingredient in our approach, which is described in Section 4, is a linear-time procedure **SolveSDP** for solving SDP programs of the form (6). As discussed above, generic SDP solvers are not suitable for this task as they are not able to exploit the sparsity of the input and run in super-linear time. In order to approximate this problem quickly, we avoid solving the primal problem directly and instead attack the dual

problem to (6):

$$\begin{aligned} p \cdot A_1 \bullet X + (1 - p) \cdot A_2 \bullet X &< \epsilon'/2 \quad \forall X \in \mathcal{K}_{n+1}, \\ 0 \leq p &\leq 1. \end{aligned}$$

We show that this one-dimensional problem can be solved quickly via a binary search that in each iteration invokes an approximate eigenvalue oracle on a matrix of the form  $pA_1 + (1 - p)A_2$ . Moreover, whenever the primal SDP (6) is feasible, we show how to efficiently recover an approximate solution, namely a matrix  $X \in \mathcal{K}_{n+1}$  such that  $A_i \bullet X \geq \epsilon'/4$ , from the outputs of the oracle calls along the execution of the binary search. Formally, in Section 4 we prove:

**Lemma 3.** *Given access to an approximate eigenvalue oracle, with probability at least  $1 - \delta$  (over the randomization of the oracle) **SolveSDP** outputs a decomposition  $X = \sum_{i=1}^r x_i x_i^\top$  of a matrix  $X \in \mathcal{K}_{n+1}$  of rank  $r = 2$  for which  $A_i \bullet X \geq \epsilon'/4$  ( $i = 1, 2$ ), or correctly declares that (6) is infeasible. The algorithm calls the oracle at most  $O(\log(\kappa/\epsilon))$  times and can be implemented to run in total time  $\tilde{O}(N\sqrt{\kappa/\epsilon})$ .*

An important feature of our SDP solver is that it produces a solution of a very low rank, namely with  $r = 2$  (it can be shown that a rank-2 solution for formulation (6) does exist, via duality conditions). As we explain below, this would allow us to recover a rank-1 solution quickly, and in turn obtain a feasible solution to (3) in linear time. Nevertheless, we preserve full generality and present our algorithm with arbitrary rank  $r$ , as our approach may work with any other SDP solver that might produce solutions of rank higher than two.

**Rounding of SDP.** Finally, we show how to obtain an approximate solution to (3) from our solution  $X$  to the relaxation, with linear-time computations. The method we describe is based on a matrix rotation procedure, given in Algorithm 2, which is a variant of a procedure due to Sturm and Zhang (2003).

The procedure provides the following guarantee, which is proved in Section 3.1 below.

**Lemma 4.** *Given a decomposition  $X = \sum_{i=1}^r x_i x_i^\top$  of a positive semidefinite matrix  $X$  of rank  $r$  and an arbitrary matrix  $A$  with  $A \bullet X \geq a$ , **SZRotation** outputs a decomposition  $X = \sum_{i=1}^r y_i y_i^\top$  such that  $y_i^\top A y_i \geq a/r$  for all  $i \in [r]$ . The procedure runs in time  $O(Nr)$ , where  $N \geq n$  is the number of non-zero entries in  $A$ .*

In particular, for being able to recover a solution in linear time, it is essential to begin with a solution to the SDP of a constant rank—which is provided by our SDP solver. Employing this decomposition, one can compute an approximate solution to the feasibility problem (3), namely a vector  $y$  satisfying (4).

### 3.1 Analysis

In the rest of this section we prove our main theorem (Theorem 2). First, we prove Lemma 3 using the results of Section 4.

---

**Algorithm 2** SZRotation( $A, X$ )

---

**input** matrices  $A$  and  $X = \sum_{i=1}^r x_i x_i^\top$  such that  $A \bullet X \geq a$

**output** matrix  $X = \sum_{i=1}^r x_i x_i^\top$  such that  $A \bullet x_i x_i^\top \geq a/r$  for all  $i$

1: let  $a' = A \bullet X$

2: **while** there exist  $x_i, x_j$  such that  $x_i^\top A x_i > a'/r$ ,  $x_j^\top A x_j < a'/r$  **do**

3:   compute a root  $t$  of the quadratic equation

$$\left(x_i^\top A x_i - \frac{a'}{r}\right) \cdot t^2 + (2x_i^\top A x_j) \cdot t + \left(x_j^\top A x_j - \frac{a'}{r}\right) = 0 \quad (7)$$

4:   replace the vectors  $x_i, x_j$  with the vectors  $\tilde{x}_i = \frac{1}{\sqrt{t^2+1}}(tx_i + x_j)$ ,  $\tilde{x}_j = \frac{1}{\sqrt{t^2+1}}(x_i - tx_j)$

5: **end while**

6: **return**  $X = \sum_{i=1}^r x_i x_i^\top$

---

*Proof of Lemma 3.* In order to show that the lemma follows from Theorem 5 below, we have to show that for any  $c \in [0, \lambda]$ , the spectral norm of the matrices  $A_1$  and  $A_2$  defined in Eq. (5) is at most 1.

We first consider the matrix  $A_1$ . Let  $x \in \mathbb{R}^{n+1}$  be some unit vector and write  $x = (\alpha, y)$  with  $\alpha \in \mathbb{R}$  and  $y \in \mathbb{R}^n$ . Since  $x$  has unit norm,  $|\alpha| \leq 1$  and  $\|y\|^2 \leq 1$ , thus

$$\begin{aligned} 2\kappa \cdot |x^\top A_1 x| &= | -c\alpha^2 + 2\alpha b^\top y + y^\top A y | \\ &\leq c\alpha^2 + 2|\alpha| \|b\| \|y\| + \|A\|_2 \|y\|^2 \\ &\leq c + 2(\|b\| + \|A\|_2) \\ &\leq 2\lambda, \end{aligned}$$

where the last inequality follows from the fact that  $2(\|A\|_2 + \|b\|) \leq \lambda$ . Since  $\lambda \leq \kappa$  and the above applies for any unit vector  $x$ , we have shown that  $\|A_1\|_2 \leq 1$ .

Similarly, for the matrix  $A_2$  we have

$$2\kappa \cdot |x^\top A_2 x| = | -\alpha^2 + y^\top M y | \leq 1 + \|M\|_2 \leq 2\lambda$$

for any unit vector  $x$ , which implies that  $\|A_2\|_2 \leq 1$ . □

Next, we prove Lemma 4.

*Proof of Lemma 4.* First, note that equation (7) has real roots since  $x_i^\top A x_i - a'/r > 0$  and  $x_j^\top A x_j - a'/r < 0$ . One can verify that  $\tilde{x}_i \tilde{x}_i^\top + \tilde{x}_j \tilde{x}_j^\top = x_i x_i^\top + x_j x_j^\top$ , so that the equality  $X = \sum_{i=1}^r x_i x_i^\top$  remains true along the execution of the algorithm. On the other hand, note

that

$$\begin{aligned}
\tilde{x}_i^\top A \tilde{x}_i &= \frac{(tx_1 + x_2)^\top A (tx_1 + x_2)}{t^2 + 1} \\
&= \frac{x_1^\top A x_1 \cdot t^2 + 2x_1^\top A x_2 \cdot t + x_2^\top A x_2}{t^2 + 1} \\
&= \frac{a'}{r},
\end{aligned}$$

where the final equality follows from  $t$  being a root of (7). Hence, each iteration produces an additional index  $i$  for which  $x_i^\top A x_i = a'/r$ , and after at most  $r$  iterations it must be the case that  $x_i^\top A x_i = a'/r \geq a/r$  for all  $i$ . Consequently, the total runtime is  $O(Nr)$  as each iteration needs  $O(N)$  time.  $\square$

We can now prove our main theorem.

*Proof of Theorem 2.* We first prove the runtime guarantee, and then show the correctness of the algorithm.

**Running time.** Note that the number of non-zero entries in each of the matrices  $A_1, A_2$  is  $O(N)$ . Hence, according to Lemma 3, the call to **SolveSDP** in line 2 of the algorithm invokes the approximate eigenvalue oracle at most  $O(\log(\kappa/\epsilon))$  times and returns in time  $\tilde{O}(N\sqrt{\kappa/\epsilon})$ . Whenever the SDP relaxation is feasible **SolveSDP** produces a solution of rank  $r = 2$ , in which case Lemma 4 states that the invocation of **SZRotation** in line 6 runs in time  $O(N)$ . Therefore, the total runtime of the entire algorithm is  $\tilde{O}(N\sqrt{\kappa/\epsilon})$ .

**Correctness.** First, assume that the algorithm returns “infeasible”. By the guarantees of **SolveSDP** (Lemma 3), this means that the SDP relaxation (6) is infeasible. Hence, the problem (3) is also infeasible (otherwise, as explained above, we could convert a feasible solution  $x$  of (3) to a feasible solution  $X$  to the SDP relaxation), as required in this case.

Next, assume that the algorithm returns a solution vector  $x \in \mathbb{R}^n$ . In this case, **SolveSDP** returns a matrix  $X$  such that  $A_i \bullet X \geq \epsilon'/4$  ( $i = 1, 2$ ). Hence, by Lemma 4, the output  $X = \sum_{i=1}^r y_i y_i^\top$  of **SZRotation** has  $y_i^\top A_1 y_i \geq \epsilon'/4r$  for all  $i = 1, 2, \dots, r$ . On the other hand, since  $A_2 \bullet X \geq \epsilon'/4$ , at least one of the vectors  $y_i$  must satisfy  $y_i^\top A_2 y_i \geq \epsilon'/4r$ . Hence, the algorithm can indeed find a vector  $y \in \{x_1, \dots, x_r\}$  such that  $y^\top A_2 y \geq \epsilon'/4r$ , for which we also have  $y^\top A_1 y \geq \epsilon'/4r$ . Rewriting the last two inequalities in terms of  $\alpha$  and  $\tilde{y}$ , we get

$$\tilde{y}^\top A \tilde{y} + 2\alpha b^\top \tilde{y} - c\alpha^2 \geq \epsilon'/4r \quad \text{and} \quad \alpha^2 - \|\tilde{y}\|_M^2 \geq \epsilon'/4r.$$

The second inequality implies that  $\alpha \neq 0$ , and so we can divide through by  $\alpha^2$ . Note that since  $X \in \mathcal{K}_{n+1}$  we also have  $\alpha^2 \leq \alpha^2 + \|\tilde{y}\|^2 = \|y\|^2 \leq \text{tr}(X) \leq 1$ , which implies that the vector  $x = \tilde{y}/\alpha$  has

$$x^\top A x + 2b^\top x - c \geq \epsilon'/4r \quad \text{and} \quad 1 - \|x\|_M^2 \geq \epsilon'/4r.$$



This in particular means that

$$x^\top Ax + 2b^\top x \geq c \quad \text{and} \quad \|x\|_M^2 \leq 1,$$

as required.  $\square$

## 4 Solving the Relaxation in Linear Time

In this section we describe a linear-time algorithm for approximately solving an SDP problem of the form

$$\begin{aligned} A_i \bullet X &\geq \epsilon \quad (i = 1, 2) \\ X &\in \mathcal{K}_n, \end{aligned} \tag{8}$$

where  $A_i \in \mathbb{R}^{n \times n}$ ,  $\|A_i\|_2 \leq 1$  ( $i = 1, 2$ ) and the number of non-zero entries in each of the matrices  $A_1, A_2$  is at most  $N \geq n$ . Namely, the algorithm we present either finds a matrix  $X \in \mathcal{K}_n$  for which  $A_i \bullet X \geq \epsilon/2$  ( $i = 1, 2$ ), or correctly declares that (8) is infeasible.

Our algorithm, given in Algorithm 3, applies as a first step a binary search for approximately solving the dual feasibility problem:

$$\begin{aligned} A(p) \bullet X &< \epsilon/2 \quad \forall X \in \mathcal{K}_n, \\ 0 &\leq p \leq 1, \end{aligned}$$

where we denote  $A(p) = pA_1 + (1-p)A_2$  for all  $p \in [0, 1]$ . The binary search either correctly declares that the dual problem is infeasible or finds  $p \in [0, 1]$  such that  $A(p) \bullet X < \epsilon$  for all  $X \in \mathcal{K}_n$ . Infeasibility of the dual implies feasibility of the primal problem (8), in which case our algorithm is able to efficiently recover a primal solution from the binary search iterates by solving a simple linear program.

The algorithm assumes the availability of an approximate eigenvector computation oracle, denoted **ApproxEV**, as described in Definition 1. In Section 4.1 we explain how such an oracle can be implemented in time  $\tilde{O}(N/\sqrt{\epsilon})$ , where  $N$  is the number of non-zero entries in the input matrix and  $\epsilon$  is the error parameter.

We now state the main result of this section.

**Theorem 5.** *Given matrices  $A_1, A_2 \in \mathbb{R}^{n \times n}$  with  $\|A_i\|_2 \leq 1$  and parameters  $\epsilon, \delta > 0$ , with probability at least  $1 - \delta$  **SolveSDP** outputs a matrix  $X \in \mathcal{K}_n$  of rank 2 that satisfies  $A_i \bullet X \geq \epsilon/2$  ( $i = 1, 2$ ), or correctly declares that (8) is infeasible. The algorithm calls the oracle **ApproxEV** at most  $O(\log(1/\epsilon))$  times and can be implemented to run in total time  $\tilde{O}(N/\sqrt{\epsilon})$ .*

For proving Theorem 5 we need two simple duality results.

**Lemma 6.** *Let  $\mathcal{S}$  be an arbitrary compact set of matrices. Exactly one of the following statements holds:*

---

**Algorithm 3** SolveSDP ( $A_1, A_2, \epsilon$ )

---

**input** matrices  $A_1, A_2 \in \mathbb{R}^n$  with  $\|A_i\|_2 \leq 1$  and  $\epsilon > 0$

**output** matrix  $X \in \mathcal{K}_n$  such that  $A_i \bullet X \geq \epsilon/2$  ( $i = 1, 2$ ), or infeasibility of (8)

```
1: initialize  $T \leftarrow \log_2(8/\epsilon)$ ,  $p_1 \leftarrow 1, p_2 \leftarrow 1$ 
2: for  $t = 1$  to  $T$  do
3:   let  $p \leftarrow (p_1 + p_2)/2$ 
4:   invoke  $(\lambda, x) \leftarrow \text{ApproxEV}(pA_1 + (1-p)A_2, \epsilon/4, \delta/T)$ 
5:   if  $\lambda < 3\epsilon/4$  then
6:     return “infeasible”
7:   else if  $x^\top A_1 x < x^\top A_2 x$  then
8:     update  $p_1 \leftarrow p, x_1 \leftarrow x$ 
9:   else {if  $x^\top A_1 x > x^\top A_2 x$ }
10:    update  $p_2 \leftarrow p, x_2 \leftarrow x$ 
11:   end if
12: end for
13: compute  $0 \leq q \leq 1$  such that
```

$$q \cdot x_1^\top A_i x_1 + (1-q) \cdot x_2^\top A_i x_2 \geq \epsilon/2 \quad (i = 1, 2) \quad (9)$$

```
14: return  $X = q \cdot x_1 x_1^\top + (1-q) \cdot x_2 x_2^\top$ 
```

---

1. There exists  $X \in \text{conv } \mathcal{S}$  such that  $A_i \bullet X \geq \epsilon$  ( $i = 1, 2$ ).

2. There exist  $p \in [0, 1]$  such that  $A(p) \bullet X = (pA_1 + (1-p)A_2) \bullet X < \epsilon$  for all  $X \in \mathcal{S}$ .

*Proof.* First assume that the first statement holds true, i.e. there exists  $X^* \in \text{conv } \mathcal{S}$  for which  $A_i \bullet X^* \geq \epsilon$  ( $i = 1, 2$ ). Then, for all  $p \in [0, 1]$  we have

$$A(p) \bullet X^* = p(A_1 \bullet X^*) + (1-p)(A_2 \bullet X^*) \geq \epsilon$$

which proves that the second statement cannot hold. Conversely, if there does not exist such  $X^*$ , then  $\min_{p \in [0, 1]} A(p) \bullet X = \min_i A_i \bullet X < \epsilon$  for all  $X \in \text{conv } \mathcal{S}$ . Applying Sion's minimax theorem (Sion, 1958), we obtain

$$\min_{p \in [0, 1]} \max_{X \in \text{conv } \mathcal{S}} A(p) \bullet X = \max_{X \in \text{conv } \mathcal{S}} \min_{p \in [0, 1]} A(p) \bullet X < \epsilon.$$

This means that there exists  $p^* \in [0, 1]$  such that  $A(p^*) \bullet X < \epsilon$  for all  $X \in \mathcal{S}$ , that is, the second statement is true.  $\square$

When  $\mathcal{S} = \{X_1, X_2, \dots, X_m\}$  is a finite set, we obtain the following corollary that enables us to compute an approximate primal solution from the binary search iterates.

**Corollary 7.** *Let  $X_1, X_2, \dots, X_m$  be arbitrary matrices and assume that there does not exist  $p \in [0, 1]$  such that  $A(p) \bullet X_i < \epsilon/2$  for all  $i = 1, 2, \dots, m$ . Then there exists  $q \in \Delta_m$  such that for  $X = \sum_{i=1}^m q_i X_i$  it holds that  $A_i \bullet X \geq \epsilon/2$  ( $i = 1, 2$ ).*

We can now provide a proof of Theorem 5.

*Proof of Theorem 5.*

**Running time.** The algorithm invokes **ApproxEV** at most  $T = O(\log(1/\epsilon))$  times on matrices of the form  $A(p)$ . Since by the triangle inequality  $\|A(p)\|_2 \leq p\|A_1\|_2 + (1-p)\|A_2\|_2 \leq 1$  (as we assume that  $\|A_1\|_2, \|A_2\|_2 \leq 1$ ), Lemma 8 shows that this procedure can be implemented using the Lanczos method to run in  $\tilde{O}(N/\sqrt{\epsilon})$ . Hence, the entire algorithm runs in  $\tilde{O}(N/\sqrt{\epsilon})$  time as all other operations are linear in the problem dimensions.

**Correctness.** First note that since each call to the oracle **ApproxEV** errs with probability at most  $\delta/T$ , with probability at least  $1 - \delta$  all oracle invocations return a correct output. Observe that if the algorithm declares that the problem is infeasible, then there exists a value of  $p$  for which **ApproxEV** outputs  $\lambda < 3\epsilon/4$ . This means that

$$\max_{X \in \mathcal{K}_n} A(p) \bullet X = \lambda_{\max}(A(p)) \leq \lambda + \epsilon/4 < \epsilon,$$

that is,  $A(p) \bullet X < \epsilon$  for all  $X \in \mathcal{K}_n$ . Lemma 6 then implies that (8) is indeed infeasible. Thus, we henceforth assume that all invocations of **ApproxEV** returned  $\lambda \geq 3\epsilon/4$ .

Consider the values of  $p_1, p_2$  and  $x_1, x_2$  at the end of the main loop of the algorithm, and denote  $X_i = x_i x_i^\top$  ( $i = 1, 2$ ). Then according to our assumption, we have  $A(p_1) \bullet X_1 \geq 3\epsilon/4$  and  $A(p_2) \bullet X_2 \geq 3\epsilon/4$ . Also, we have  $p_2 - p_1 \leq \epsilon/8$  since the binary search continues for  $\log_2(8/\epsilon)$  iterations. Our central observation is that the system of two inequalities  $A(p) \bullet X_i < \epsilon/2$  ( $i = 1, 2$ ) must be infeasible (in  $p$ ). Indeed, assume that there exists some  $p^* \in [0, 1]$  for which  $A(p^*) \bullet X_1 < \epsilon/2$  and  $A(p^*) \bullet X_2 < \epsilon/2$ . Notice that  $A(p) \bullet X_1 \geq 3\epsilon/4$  for  $p < p_1$ , as  $A(p_1) \bullet X_1 \geq 3\epsilon/4$  and the function  $p \mapsto A(p) \bullet X_1$  is monotonically decreasing (recall that  $x_1^\top A_1 x_1 < x_1^\top A_2 x_1$ ). Similarly we have  $A(p) \bullet X_2 \geq 3\epsilon/4$  for  $p > p_2$ , so it must be the case that  $p^* \in [p_1, p_2]$ . But this implies that

$$A(p_1) \bullet X_1 - A(p^*) \bullet X_1 = (p^* - p_1) \cdot (A_2 \bullet X_1 - A_1 \bullet X_1) \leq (p_2 - p_1) \cdot 2 \leq \epsilon/4$$

from which we get that  $A(p^*) \bullet X_1 \geq A(p_1) \bullet X_1 - \epsilon/4 \geq \epsilon/2$  which is a contradiction to the choice of  $p^*$ .

This infeasibility, together with Corollary 7, leads to the conclusion that there exists  $q \in [0, 1]$  such that  $X = qX_1 + (1 - q)X_2 \in \mathcal{K}_n$  satisfies  $A_i \bullet X \geq \epsilon/2$  ( $i = 1, 2$ ). Since the existence of such  $q$  is guaranteed, it can be found by solving the simple linear program (9), thereby retrieving a rank-two matrix which is an approximate solution to (8).  $\square$

## 4.1 Approximate Eigenvalue Computation

In our analysis we require a linear-time procedure for approximating eigenvalues of sparse matrices. The following lemma states that the *Lanczos method* provides that.

**Lemma 8.** *There exists an algorithm that given a matrix  $M \in \mathbb{R}^{n \times n}$  with  $\|M\|_2 \leq \lambda$  and parameters  $\epsilon, \delta > 0$ , runs in time*

$$O\left(\frac{N\sqrt{\lambda}}{\sqrt{\epsilon}} \ln \frac{n}{\delta}\right)$$

*where  $N$  is the number of non-zero entries in the matrix  $M$ , and returns a unit vector  $x \in \mathbb{R}^n$  for which  $x^\top Mx \geq \lambda_{\max}(M) - \epsilon$  with probability at least  $1 - \delta$ .*

The proof relies on the analysis of the Lanczos method provided by Kuczynski and Wozniakowski (1992).

*Proof.* The statement of the lemma is proved in Theorem 4.2 of Kuczynski and Wozniakowski (1992) when  $M$  is a positive semidefinite matrix with  $\|M\|_2 \leq 1$ . To prove the lemma for an arbitrary matrix  $M$  with  $\|M\|_2 \leq \lambda$ , consider the matrix  $M' = \frac{1}{2\lambda}M + \frac{1}{2}I$  which is positive semidefinite with  $\|M'\|_2 \leq 1$ . If we apply the Lanczos method with error parameter  $\epsilon' = \frac{1}{2\lambda}\epsilon$ , we obtain with high probability a unit vector  $x$  such that  $x^\top M'x \geq \lambda_{\max}(M') - \epsilon'$ . Hence,

$$\frac{1}{2\lambda}x^\top Mx + \frac{1}{2} = x^\top M'x \geq \lambda_{\max}(M') - \epsilon' = \frac{1}{2\lambda}\lambda_{\max}(M) + \frac{1}{2} - \frac{1}{2\lambda}\epsilon,$$

so that  $x^\top Mx \geq \lambda_{\max}(M) - \epsilon$ , as required.  $\square$

## References

- F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, 5(1):13–51, 1995.
- S. Arora, E. Hazan, and S. Kale. Fast algorithms for approximate semidefinite programming using the multiplicative weights update method. In *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pages 339–348. IEEE, 2005.
- A. Ben-Tal and M. Teboulle. Hidden convexity in some nonconvex quadratically constrained quadratic programming. *Mathematical Programming*, 72(1):51–63, 1996.
- S. Busygin. A new trust region technique for the maximum weight clique problem. *Discrete Applied Mathematics*, 154(15):2080–2096, 2006.
- A. R. Conn, N. I. Gould, and P. L. Toint. *Trust region methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.
- J. B. Erway and P. E. Gill. A subspace minimization method for the trust-region step. *SIAM Journal on Optimization*, 20(3):1439–1461, 2009.
- J. B. Erway, P. E. Gill, and J. D. Griffin. Iterative methods for finding a trust-region step. *SIAM Journal on Optimization*, 20(2):1110–1131, 2009.

- W. Gander, G. H. Golub, and U. von Matt. A constrained eigenvalue problem. *Linear Algebra and its applications*, 114:815–839, 1989.
- N. I. Gould, S. Lucidi, M. Roma, and P. L. Toint. Solving the trust-region subproblem using the lanczos method. *SIAM Journal on Optimization*, 9(2):504–525, 1999.
- N. I. Gould, D. P. Robinson, and H. S. Thorne. On solving trust-region and other regularised subproblems in optimization. *Mathematical Programming Computation*, 2(1):21–57, 2010.
- J. Kuczynski and H. Wozniakowski. Estimating the largest eigenvalue by the power and lanczos algorithms with a random start. *SIAM journal on matrix analysis and applications*, 13(4):1094–1122, 1992.
- J. J. Moré and D. C. Sorensen. Computing a trust region step. *SIAM Journal on Scientific and Statistical Computing*, 4(3):553–572, 1983.
- Y. Nesterov. A method of solving a convex programming problem with convergence rate  $O(1/k^2)$ . *Soviet Mathematics Doklady*, 27(2):372–376, 1983.
- Y. Nesterov, A. S. Nemirovskii, and Y. Ye. *Interior-point polynomial algorithms in convex programming*, volume 13. SIAM, 1994.
- C. H. Papadimitriou and T. Roughgarden. Computing correlated equilibria in multi-player games. *Journal of the ACM (JACM)*, 55(3):14, 2008.
- F. Rendl and H. Wolkowicz. A semidefinite framework for trust region subproblems with applications to large scale minimization. *Mathematical Programming*, 77(1):273–299, 1997.
- M. Rojas, S. A. Santos, and D. C. Sorensen. A new matrix-free algorithm for the large-scale trust-region subproblem. *SIAM Journal on optimization*, 11(3):611–646, 2001.
- M. Sion. On general minimax theorems. *Pacific Journal of Mathematics*, 8(1):171–176, 1958.
- D. Sorensen. Minimization of a large-scale quadratic function subject to a spherical constraint. *SIAM Journal on Optimization*, 7(1):141–161, 1997.
- J. F. Sturm and S. Zhang. On cones of nonnegative quadratic functions. *Mathematics of Operations Research*, 28(2):246–267, 2003.
- Y. Ye and S. Zhang. New results on quadratic minimization. *SIAM Journal on Optimization*, 14(1):245–267, 2003.
- H. Zhang, A. R. Conn, and K. Scheinberg. A derivative-free algorithm for least-squares minimization. *SIAM Journal on Optimization*, 20(6):3555–3576, 2010.